UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/628,054 | 07/25/2003 | Vinod K. Grover | 3382-65598-01 | 4135 |

26119 7590 10/30/2007

KLARQUIST SPARKMAN LLP
121 S.W. SALMON STREET
SUITE 1600
PORTLAND, OR 97204

| EXAMINER | |
|---|---|
| TECKLU, ISAAC TUKU | |
| **ART UNIT** | **PAPER NUMBER** |
| 2192 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 10/30/2007 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 10/628,054 | GROVER ET AL. | |
| | Examiner | Art Unit | |
| | Isaac T. Tecklu | 2192 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *25 July 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-43* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-43* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *See Continuation Sheet*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

Continuation of Attachment(s) 3). Information Disclosure Statement(s) (PTO/SB/08), Paper No(s)/Mail Date :11/08/06, 07/10/06, 05/05/06, 12/23/05, 10/14/05, 07/30/04, .

## DETAILED ACTION

1. This action is responsive to the application filed on 7/25/2003.

2. Claims 1-43 have been examined.

### *Oath/Declaration*

3. The office acknowledges receipt of a properly signed oath/declaration filed on 7/25/2003.

### *Specification*

4. The use of trademark *Microsoft* has been noted in this application (page 8, lines 20-25, etc). It should be capitalized wherever it appears and be accompanied by the generic terminology. The trademark is accompanied by the acronym *CLR*. However, this acronym is undefined so it is not known whether this acronym provides the aforementioned generic terminology.

Although the use of trademark is permissible in patent applications, the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner which might adversely affect their validity as trademarks.

5. Status of related applications on page 1 of the specification should be updated.

### *Claim Objections*

6. Claims 2, 14 and 28-29, recite acronym "JIT", such acronym should be spelled out once in the claims as its intended meaning and utility will be changed over time. Appropriate correction is required.

7. Claim 28 recites acronym "CDK", such acronym should be spelled out once in the claims as its intended meaning and utility will be changed over time. Appropriate correction is required.

### *Claim Rejections - 35 USC § 103*

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person

having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 1-43 are rejected under 35 U.S.C. 103(a) as being unpatentable over "The Jalepeno Dynamic Optimizing Compiler for Java by Burke et al. (hereinafter "Burke") in view of Bates et al. (hereinafter "Bates").

Per claim 1, Burke discloses one or more computer-readable media with computer-executable instructions for implementing a software development architecture comprising:

a software development scenario-independent intermediate representation format (e.g. Figure 3, "HIR" and related text);

one or more exception handling models operable to support a plurality of programming language specific exception handling models (e.g. page 131, section 2 of section 4);

a code generator operable to generate code targeted for a plurality of execution architectures (e.g. Figure 3, "Binary Code" and related text).

Burke does not explicitly disclose a type system operable to represent the type representations of a plurality of source languages. However Bates discloses the code transformation program 130 transforms the source code into the intermediate code, alternatively a compiler that is adapted to analyze byte codes such as the optimizing Java 119 may incorporate the code transformation program 130 to optimize compilation operations such as copy propagation in paragraph [0031]. Bates discloses the source code to comprise one or more programs or files that is generally written in a programming language such as C,C++, Pascal, Java and the like in paragraph [0028]. Therefore, it would have been obvious to one skilled in the art at the time of the invention was made to combine Burke and Bates a type system operable to represent the type representations of a plurality of source languages to allow programmer to more easily develop and debug the source code during software development as once suggested by Bates in paragraph [0030].

Per claim 2, Burke discloses the one or more computer-readable media of claim 1 wherein the architecture is scalable to produce target software development tools (e.g. Figure 3, "Binary Code" and related text) ranging from lightweight JIT compilers (e.g. page 139, section

11 "... just-in-time compilers ...") to whole program optimizing compilers (e.g. page 139, section 11 "... simple optimization ... generate target machine code...").

Per claim 3, Burke discloses the one or more computer-readable media of claim 1 wherein the architecture can be configured to produce a target software development tool with varying ranges of memory footprint (e.g. page 133, section 6.2 "... memory true/anti/output dependences ...'), compilation speed (e.g. page 139, section 11, "... fast dynamic compilation ..."), and optimization (e.g. page 139, section 11, "... selectively optimize ...").

Per claim 4, Burke discloses the one or more computer-readable media of claim 1 wherein the software development architecture is operable to produce a software development tool modifiable by combining a modification component with the software development architecture (e.g. page 135, section 6.6 "... HIR result in the modifications ...").

Per claim 5, Burke discloses the one or more computer-readable media of claim 1 wherein the software development architecture is operable to produce a software development tool by dynamically linking a binary version of the software development architecture to a modification component (e.g. page 130, section 2 "... dynamic linker ...").

Per claim 6, Burke discloses the one or more computer-readable media of claim 1 wherein the intermediate representation format is extensible at runtime of a software tool employing the intermediate representation format (e.g. page 130, section 2 "... dynamic compilers, and support for other run-time features ...").

Per claim 7, Burke discloses the one or more computer-readable media of claim 1 wherein the architecture is combinable with one or more software development components (e.g. Figure 1 and related text).

Per claim 8, Burke discloses the one or more computer-readable media of claim 7 wherein the one or more software development components comprise data describing a target software development tool (e.g. page 130, section 3, "optimization plan").

Per claim 9, Burke discloses the one or more computer-readable media of claim 7 wherein the one or more software development components provides target execution architecture data to the code generator (e.g. page 130, section 3, "optimization plan").

Per claim 10, Burke discloses the one or more computer-readable media of claim 7 wherein the one or more software development components provide one or more type-checking rules to the type system (e.g. page 134, section 6.3 "grammar rule ").

Per claim 11, Burke discloses the one or more computer-readable media of claim 7 wherein the one or more software development components provide a set of class extension declarations to the architecture (e.g. Figure 5 and related text).

Per claim 12, Burke discloses the one or more computer-readable media of claim 7 wherein the combined one or more software development components and architecture produce a target software development tool (e.g. Figure 1 and related text).

Per claim 13, Burke discloses the one or more computer-readable media of claim 12 wherein the target software development tool comprises a native compiler (e.g. Figure 1 "optimizing compiler " and related text).

Per claim 14, Burke discloses the one or more computer-readable media of claim 12 wherein the target software development tool comprises a JIT compiler (e.g. page 139, section 11 "... just-in-time compilers ... simple optimization ... generate target machine code...").

Per claim 15, Burke discloses a method of creating a target software development tool, the method comprising:

receiving at least one computer-readable specification specifying functionality specific to one or more software development scenarios (e.g. Figure 3, "HIR" and related text);

creating at least one software development component from the at least one specification; and integrating the at least one software development component into a software development scenario-independent framework (e.g. Figure 3, "Binary Code" and related text).

Per claim 16, Burke discloses the method of claim 15 further comprising: compiling the at least one software development component and framework to create the target software development tool (e.g. page 130, section 3, "optimization plan").

Per claim 17, Burke discloses the method of claim 15 wherein software development components created from a plurality of computer-readable specifications for a plurality of respective software development scenarios are integrated into the framework (e.g. Figure 3, "HIR" and related text).

Per claim 18, Burke discloses the method of claim 17 wherein the plurality of computer-readable specifications specify functionality for the following respective software development scenarios: target execution architecture; input language or input binary format (e.g. Figure 3, "Byte code to HIR" and related text); and compilation type (e.g. Figure 2, "Opt-compiled Code").

Per claim 19, Burke discloses the method of claim 15 wherein the computer-readable specification specifies functionality for target execution architecture of the software development tool (e.g. Figure 3, "Profile Information" and related text).

Per claim 20, Burke discloses the method of claim 15 wherein the computer-readable specification specifies functionality for accommodating an input language for the software development tool (e.g. Figure 3, "Byte code to HIR" and related text).

Per claim 21, Burke discloses the method of claim 15 wherein the computer-readable specification specifies functionality for accommodating a binary input for the software development tool (e.g. Figure 3, "Byte code to HIR" and related text).

Per claim 22, Burke discloses the method of claim 15 wherein the computer-readable specification comprises one or more rule sets for type-checking one or more languages (e.g. page 134, section 6.3 "grammar rule ").

Per claim 23, Burke discloses the method of claim 15 wherein the computer-readable specification comprises a set of class extension declarations specific to one or more of the software development scenarios (e.g. Figure 5 and related text).

Per claim 24, Burke discloses the method of claim 15 wherein the computer-readable specification comprises functionality for processing an intermediate representation format capable of representing a plurality of programming languages (e.g. page 132, section 2 "... Java source program ...").

Per claim 25, Burke discloses the method of claim 24 wherein the intermediate representation format comprises one or more exception handling models capable of supporting a plurality of programming language-specific exception handling models (e.g. page 131, section 2 of section 4).

Per claim 26, Burke discloses the method of claim 24 wherein the intermediate representation comprises type representations capable of representing the type representations of a plurality of programming languages (e.g. page 131, section 2 of section 4).

Per claim 27, Burke discloses the method of claim 15 further comprising: integrating custom code specific to one of the software development scenarios (e.g. Figure 2 and related text).

Per claim 28, Burke discloses the method of claim 15 wherein the software development tool comprises one of the group consisting of: a native compiler, a JIT compiler, an analysis

tool, and a CDK (e.g. page 139, section 11 "... just-in-time compilers ... simple optimization ... generate target machine code...").

Per claim 29, Burke discloses the method of claim 15 wherein the computer-readable specification specifies functionality of one of the group consisting of: a Pre-JIT compiler functionality, optimizer functionality, and defect detection tool functionality (e.g. page 139, section 11 "... just-in-time compilers ... simple optimization ... generate target machine code...").

Per claim 30, Burke discloses one or more computer-readable media containing one or more computer-executable instructions for performing the method of claim 15 (e.g. Figure 3, "HIR" and related text).

Per claim 31, Burke discloses a method of creating a target software development tool from a common framework, the method comprising: configuring the common framework based on one or more characteristics of the target software development tool;

integrating data comprising one or more characteristics of the target software development tool into the common framework (e.g. Figure 2 and related text); and

creating the target software development tool from the integrated common framework (e.g. Figure 3, "Binary Code" and related text).

Per claim 32, Burke discloses the method of claim 31 wherein the one or more characteristics can comprise the amount of memory necessary for the target software development tool to execute on a target architecture, the speed at which the target software development tool will execute on a target architecture (e.g. page 139, section 11, "... fast dynamic compilation ..."), an input language for the target software development tool, a input binary format for the target software development tool, or the target architecture for the target software development tool to execute on a target architecture (e.g. Figure 3, "Byte code to HIR" and related text).

Per claim 33, Burke discloses a method of creating a software development tool, the method comprising: receiving at least one computer-readable specification specifying:

a type of software development tool to be created (e.g. Figure 3, "Binary Code" and related text),

functionality for a target execution architecture for the software development tool (e.g. Figure 3, "Byte code to HIR" and related text),

functionality for accommodating an input language for the software development tool (e.g. Figure 3, "Byte code to HIR" and related text), and

wherein the at least one computer-readable specifications comprises a set of class extension declarations (e.g. Figure 5 and related text);

integrating the at least one computer-readable specification into a software development architecture (e.g. Figure 3, "Binary Code" and related text); and

creating the software development tool via the integrated specification and architecture, wherein the software development tool is of the type specified, is targeted to the target execution architecture specified, accommodates the input language specified, and processes the intermediate representation (e.g. Figure 3, "Binary Code" and related text).

Burke does not explicitly disclose functionality for processing an intermediate representation capable of representing a plurality of programming languages. However Bates discloses the code transformation program 130 transforms the source code into the intermediate code, alternatively a compiler that is adapted to analyze byte codes such as the optimizing Java 119 may incorporate the code transformation program 130 to optimize compilation operations such as copy propagation in paragraph [0031]. Bates discloses the source code to comprise one or more programs or files that is generally written in a programming language such as C,C++, Pascal, Java and the like in paragraph [0028]. Therefore, it would have been obvious to one skilled in the art at the time of the invention was made to combine Burke and Bates a type system operable to represent the type representations of a plurality of source languages to allow programmer to more easily develop and debug the source code during software development as once suggested by Bates in paragraph [0030].

Per claim 34, Burke discloses a method of producing inter-compatible software development tools, the method comprising:

creating a first software development tool from a software development architecture (e.g. Figure 3, "Binary Code" and related text); and

creating a second software development tool based on the first software development tool, wherein the second software development tool dynamically links to a binary version of the software development architecture (e.g. Figure 2, "Opt-compiled Code" and related text).

Per claim 35, Burke discloses the method of claim 34 wherein the binary version of the software development architecture contains classes that are extensible through a set of declarations (e.g. Figure 5 and related text).

Per claim 36, Burke discloses the method of claim 34 wherein the software development architecture comprises functionality for an intermediate representation format used by both the first and second software development tools (e.g. page 131, section 2 of section 4).

Per claim 37, Burke discloses the method of claim 34 wherein the software development architecture comprises functionality for a type system used by both the first and second software development tools (e.g. Figure 2 and 3 and related text).

Per claim 38, Burke discloses the method of claim 34 wherein the software development architecture comprises functionality for exception handling models used by both the first and second software development tools (e.g. page 131, section 2 of section 4).

Per claim 39, Burke discloses a method of modifying a software development tool, the software development tool having been created using a software development architecture comprising one or more software development components, the method comprising:

dynamically linking a software development component not present in the software development architecture to a binary version of the software development architecture (e.g. page 130, section 2 "... dynamic linker ..."); and

creating a modified software development tool from the dynamically linked binary version and the software development component (e.g. Figure 3, "Binary Code" and related text).

Per claim 40, Burke discloses the method of claim 39 wherein the binary version of the software development architecture comprises classes that are extensible through a set of declarations (e.g. Figure 5 and related text).

Per claim 41, Burke discloses the method of claim 39 wherein the binary version of the software development architecture comprises functionality for a type system used by the modified software development tool (e.g. page 135, section 6.6 "... HIR result in the modifications ...").

Per claim 42, Burke discloses the method of claim 39 wherein the binary version of the software development architecture comprises functionality for exception handling models used by the modified software development tool.

Per claim 43, Burke discloses a method of creating a software development tool, the method comprising:

receiving at least one computer-executable file comprising:

one or more exception handling models capable of supporting a plurality of programming language specific exception handling models (e.g. page 131, section 2 of section 4);

a type system capable of representing the type representations of a plurality of source languages (e.g. Figure 3, "Optimized MIR" and related text); and a code generator capable of generating code targeted for a plurality of execution architectures (e.g. Figure 3, "Binary Code" and related text);

linking a software component to the at least one computer-executable file using at least one class extension declaration (e.g. page 130, section 2 "... dynamic linker ..."); and

creating the software development tool via the linked software component and computer-executable file (e.g. Figure 3, "Binary Code" and related text).

Burke does not explicitly disclose an intermediate representation capable of representing a plurality of programming languages and computer executable images. However Bates discloses the code transformation program 130 transforms the source code into the intermediate

code, alternatively a compiler that is adapted to analyze byte codes such as the optimizing Java 119 may incorporate the code transformation program 130 to optimize compilation operations such as copy propagation in paragraph [0031]. Bates discloses the source code to comprise one or more programs or files that is generally written in a programming language such as C,C++, Pascal, Java and the like in paragraph [0028]. Therefore, it would have been obvious to one skilled in the art at the time of the invention was made to combine Burke and Bates a type system operable to represent the type representations of a plurality of source languages to allow programmer to more easily develop and debug the source code during software development as once suggested by Bates in paragraph [0030].

***Conclusion***

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Isaac T. Tecklu whose telephone number is (571) 272-7957. The examiner can normally be reached on M-TH 9:300A - 8:00P.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

TUAN DAM
SUPERVISORY PATENT EXAMINER

Isaac Tecklu
Art Unit 2192